Create a Login Window in WPF

One of my most widely-read blog posts had to do with creating a Login Windows in WPF that I wrote several years ago. I thought I would revisit this login screen with an updated version in Visual Studio 2012 and with an updated look and feel.

Most business applications require some sort of security system. You can always use Windows Authentication to authenticate a user, but sometimes you might want your own authentication scheme. When you do, you will need to create a login screen for your user to enter her login id and password. This article will explore creating a login window in WPF.

The UI for your login screen can contain any images and controls that you would like. In Figure 1 you can see a sample login screen that has an image of a key, a large title label across the top, two labels, a text box, a password box and two button controls. You can also make the border of the window rounded so that there is no close, minimize or maximize button and no title bar.

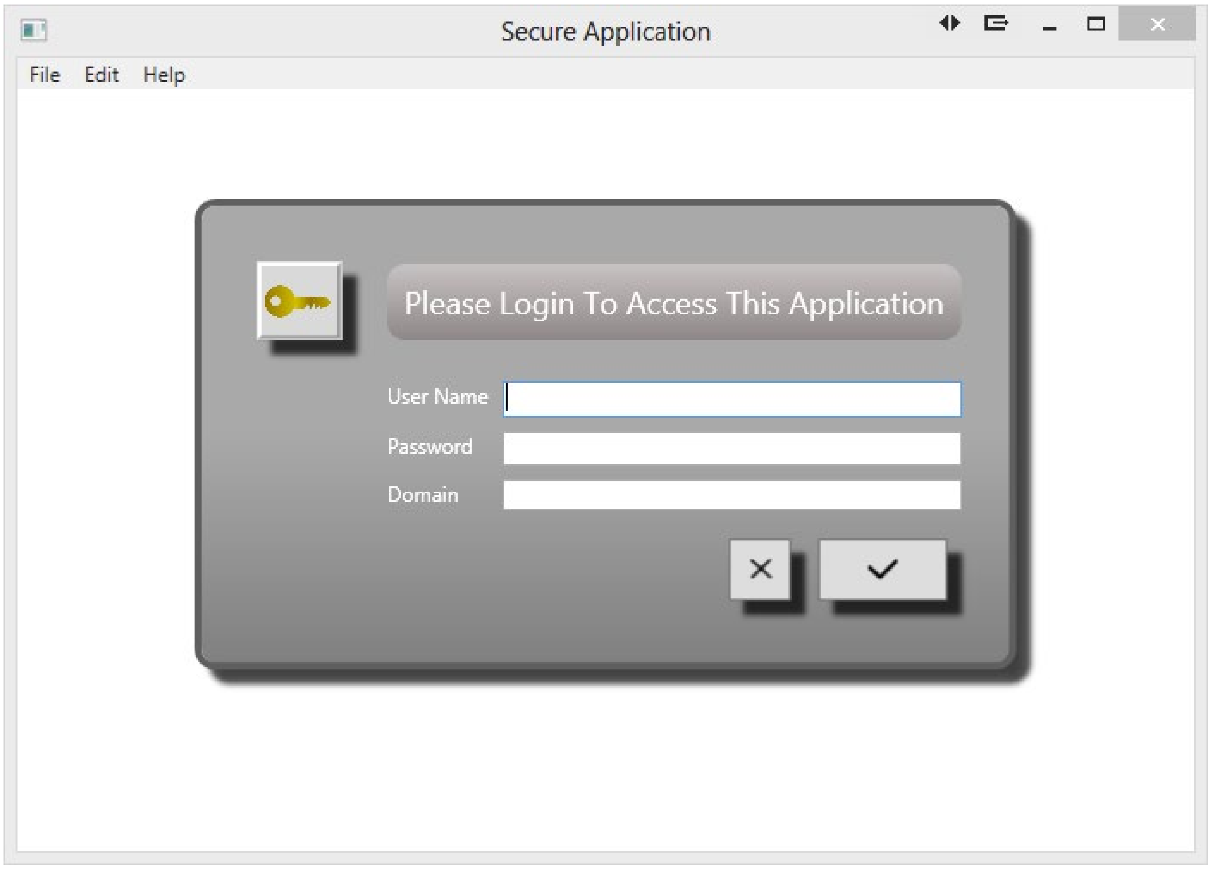

Figure 1: A Login Screen

Of course this is just one version of a login screen but let's take a look at how this is put together.

Creating the Window

To start, you need to create a window with no border and can be made into any shape you want. To do this you will set a few different attributes. The WindowStyle attribute normally allows you to set a single border, three-D border, or a Tool Window border. Setting this attribute to None will eliminate the border. The ShowInTaskbar attribute is optional, but if you are building a login screen you probably won't want this window to show up in the Task Bar as it is going to be modal style form. The next two attributes, AllowsTransparency and Background work together. You must set AllowsTransparency to True to allow the Background to be set to Transparent. If you do not set these two attributes, then your border will still show up. Below is the xaml for this window.

```
<Window ...
   WindowStartupLocation="CenterScreen"
   AllowsTransparency="True"
   ShowInTaskBar=False
   Background="Transparent"
   WindowStyle="None"
   SizeToContent="WidthAndHeight"
   FocusManager.FocusedElement=
          "{Binding ElementName=txtUserName}">

   ...
   ...

</Window>
```

There are three additional attributes that are set on this window. The WindowStartupLocation attribute is set to “CenterScreen” to ensure that the login screen is displayed in the middle of the screen when it is shown. You also set the SizeToContent attribute to WidthAndHeight to just take as much room for this window as the controls need that are contained within this window. The FocusManager.FocusedElement attribute is data-bound to the textbox control next to the User Name label. This tells WPF to place the cursor in this textbox once the screen is displayed.

The Border

Now that you have the Window xaml defined you now can create the look for the outside border of the window. A Border control is used to form the outside of this login screen. You will set the CornerRadius attribute to “10” to give the nice rounded corners. You can set the BorderBrush to “Gray” and the BorderThickness to “3”. You also want to give this border a nice wide Margin to allow room for the DropShadowEffect that we add to the outside of this border. If you do not do this, then the drop shadow will be chopped off.

The DropShadowEffect is created as a resource with a key of “shadowWindow” as shown below:

```
<Window.Resources>
  <DropShadowEffect x:Key="shadowWindow"
                    Color="Black"
                    Opacity=".75"
                    ShadowDepth="12" />
</Window.Resources>
```

When you create the Border for this window you set the Effect property to reference this static resource.

```
<Border CornerRadius="10"
        BorderBrush="#FF5F5F5F"
        BorderThickness="4"
        Background="{StaticResource backBrush}"
        Effect="{StaticResource shadowWindow}"
        Margin="24"
        Padding="24">
```

Using a Grid Layout

To place each of the login screen elements within the border, a Grid control is used with specific column and row definitions. There are three columns in this login screen. One for the image of the key, one for the labels and one for the TextBox, PasswordBox and Button controls.

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition MinWidth="80" Width="Auto" />
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>

   ...
   ...

</Grid>
```

Placing the Key Image

The Key image that is in the upper left hand corner of this login screen is placed there by using an Image control. Notice the Grid.Column, Grid.Row and Grid.RowSpan attributes that are set on the StackPanel. The Grid.Row and Grid.Column specify in which row and column of the grid you wish to display the Image control. The Grid.RowSpan allows the key to float down over the next three rows of the Grid control if necessary. If you were to use a smaller or larger key image, then you would probably need to adjust this attribute accordingly. The

Image control sets the source of its image to the KeyComputer.jpg file located in the /Images folder. A drop shadow effect is applied to this image control just like you did with the Border control.

```
<Image Grid.Column="0"
       Grid.Row="0"
       Grid.RowSpan="3"
       Effect="{StaticResource shadowWindow}"
       VerticalAlignment="Top"
       HorizontalAlignment="Left"
       Name="imgKey"
       Width="50"
       Margin="8"
       Source="Images/KeyComputer.png" />
```

The Large Title Label

The large label across the top of the login screen is simply a TextBlock control within a Border control that has the appropriate Grid.Row, Grid.Column and Grid.ColumnSpan attributes set for placement. A FontSize of 18 is applied to make the text appear larger than the other labels on this screen. A Margin of 10 is used to give us some spacing from the border of the grid.

```
<Border Grid.Column="1"
        Grid.Row="0"
        Grid.ColumnSpan="2"
        Margin="4,10,4,20"
        CornerRadius="10">
  <Border.Background>
    <LinearGradientBrush EndPoint="0.5,1"
                         StartPoint="0.5,0">
      <GradientStop Color="#FFC7C2C2"
                    Offset="0" />
      <GradientStop Color="#FF8D8787"
                    Offset="1" />
    </LinearGradientBrush>
  </Border.Background>
  <TextBlock FontSize="18"
             Margin="10"
             Text="Please Login To Access This Application" />
</Border>
```

The Login Data Controls

The controls that gather the user name and password should be fairly familiar to you if you have been doing any WPF at all. Each control is placed into a specific row and column of the Grid control. Notice the use of the Tooltip attribute on the Login TextBox, the PasswordBox and Domain TextBox control. This gives the user an idea of what to put into each control if they hover their mouse over that control.

```
<TextBlock Grid.Column="1"
            Grid.Row="1"
            Text="User Name" />
<TextBox Grid.Column="2"
          Grid.Row="1"
          ToolTipService.ToolTip="Enter Your User Name"
          Name="txtUserName" />
<TextBlock Grid.Column="1"
            Grid.Row="2"
            Text="Password" />
<PasswordBox Grid.Column="2"
              Grid.Row="2"
              ToolTipService.ToolTip="Enter Your Password"
              Name="txtPassword" />
<TextBlock Grid.Column="1"
            Grid.Row="3"
            Text="Domain" />
<TextBox Grid.Column="2"
          Grid.Row="3"
          ToolTipService.ToolTip="Enter Domain Name to Login To"
          Name="txtDomain" />
```

The Buttons

The two buttons at the bottom of the screen are placed into the last row of the Grid control and into the second column of the grid by wrapping them into a StackPanel. The StackPanel has its HorizontalAlignment attribute set to Center and its Orientation attribute to Horizontal to allow the buttons to be centered within the StackPanel and to have the buttons appear side-by-side to each other.

```
<StackPanel Grid.Column="2"
            Grid.Row="4"
            Margin="4"
            HorizontalAlignment="Right"
            Orientation="Horizontal">
  <Button Name="btnCancel"
          Click="btnCancel_Click"
          IsCancel="True"
          Effect="{StaticResource shadowWindow}"
          ToolTipService.ToolTip="Cancel">
    <Image Source="Images/XBlack.png" />
  </Button>
  <Button Name="btnLogin"
          Click="btnLogin_Click"
          IsDefault="True"
          Width="75"
          Effect="{StaticResource shadowWindow}"
          ToolTipService.ToolTip="Login">
    <Image Source="Images/CheckMarkBlack.png" />
  </Button>
</StackPanel>
```

There are two special attributes that are set on these buttons. The IsCancel attribute is set to true on the Cancel button. Setting this attribute to true will fire the click event procedure on the Cancel button if the user presses the Escape key. The IsDefault attribute is set to true on the on the Login button. Setting this attribute to true will fire the click event procedure on the Login button if the user presses the Enter key.

Writing the Code for the Login Screen

In each of the click event procedures you will need to close the screen. In the Cancel click event procedure you will set the DialogResult property of the screen to a false value. This will inform the calling procedure that the user clicked on the Cancel button on this screen. In the Login click event procedure you will set the DialogResult property of the screen to a true value. This informs the calling procedure that the user clicked on the Login button and was authenticated. I am leaving it up to you to write the code for authenticating the user. Here is the code for the Cancel event procedure.

```
private void btnCancel_Click(object sender, RoutedEventArgs e)
{
  DialogResult = false;
}
```

And, here is the code for the Login event procedure.

```
private void btnLogin_Click(object sender, RoutedEventArgs e)
{
  // Write code here to authenticate user
  // If authenticated, then set DialogResult=true
  DialogResult = true;
}
```

Displaying the Login Screen

At some point when your application launches, you will need to display your login screen modally. Below is the code that you would call to display the login form (named frmLogin in my sample application). This code is called from the main application form, and thus the owner of the login screen is set to “this”. You then call the ShowDialog method on the login screen to have this form displayed modally. After the user clicks on one of the two buttons you need to check to see what the DialogResult property was set to. The DialogResult property is a nullable type and thus you first need to check to see if the value has been set.

```
private void DisplayLoginScreen()
{
  winLogin win = new winLogin();

  win.Owner = this;
  win.ShowDialog();
  if (win.DialogResult.HasValue && win.DialogResult.Value)
    MessageBox.Show("User Logged In");
  else
    this.Close();
}
```

Summary

Creating a nice looking login screen is fairly simple to do in WPF. Using the DropShadowEffect can add a nice finished look to not only your form, but images and buttons as well. Using a border-less window is a great way to give a custom look to a login screen or splash screen. The DialogResult property on WPF Windows allows you to communicate back to the calling routine what happened on the modal screen. I hope this article gave you some ideas on how to create a login screen in WPF.

NOTE: You can download the sample code for this article by visiting my website at http://www.pdsa.com/downloads. Select “Tips & Tricks”, then select “WPF Login Screen 2013” from the drop down list.